Ramanujan - Partição de um número análise IA

Feito e Desenvolvido por Marlon Sousa

$\frac{1}{2L}\int_{-\infty}^{+\infty}|f(x)|^2dx$

Definição

Sejam n e k inteiros. Uma partição de n em k partes é uma solução integral do sistema

$n = n_1 + ... + n_k$
$1 \leq n_1 \leq ... \leq n_k$

Euler foi o primeiro a tentar resolver o desafio das partições

O número total de partições de n em s de diferentes tamanhos de peças é denotado por $t(n, s)$
Se s for especificado, então $t(n, k, s) = 0$ se $k \leq s - 1$ e $k \geq n - (\frac{s(s-1)}{2})$ ou $n \leq maxk, \frac{s(s+1)}{2}$ teremos:

$$t(n, s) = \sum_{k=s}^{\frac{2n-s(s-1)}{2}} t(n, k, s) = \sum_{k\geq 1} t(n, k, s)$$

Se $s = 1$, então $k \geq 1, n \geq k$ e

$$t(n, k, 1) = 1, se\ n\ é\ multiplo\ de\ k, 0, outro\ caso$$

Portanto

$$\sum_{n\geq} t(n, k, q)q^n = \frac{q^k}{1-q^k}$$

E além disso é fácil ver que

$t(n,2,2) = [\frac{n-1}{2}]$

Onde $[x]$ é o maior número inteiro $\leq x$. Então teremos

$\sum_{n\geq} t(n,2,2)q^2 = q^3 + q^4 + 2q^5 + 2q^6 + 3q^7 + 3q^8 + ...$

$= \frac{q^3}{1-q} + \frac{q^5}{1-q} + \frac{q^7}{1-q} + ...$

$= \frac{q^3}{(1-q)(1-q^2)}$

Identidade Principal

O objetivo desta seção é derivar uma fórmula explícita para t (n, k, 2). Antes de dar o próximo
Teorema, apresentamos algumas notações. Deixar

Nós iremos derivar a fómula explícita para $t(n,k,2)$, mas antes temos que apresentar algumas notações

- $\Phi_i(j) = 1, se\ j \equiv 0(modi); 0, caso\ outro$
- $X_k(i,j) = 0, se\ i! = 0\ e\ gcd(k,i)! = 1\ e\ gcd(i,j) = 1; 1, caso\ outro$
- $W_k = [W_k(i,j), 0 \leq i \leq k-1, 1 \leq j \leq k-1$, uma matriz cujo os elementos são dados por:
  $w_k(i,j) = d, se\ i \in I_{j,k}(d)eX_k(i,j) = 1; j, caso\ outro$

Por exemplo, para $k = 6$ nós temos.

$$W_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 4 & 4 \\ 0 & 0 & 3 & 1 & 3 \\ 0 & 2 & 0 & 4 & 2 \\ 0 & 0 & 3 & 0 & 1 \\ 0 & 2 & 3 & 4 & 0 \end{bmatrix}$$

Teorema. para $n \geq 3, n \equiv i(modk), 2 \leq k \leq n-1$, nós temos

$t(n, k, 2) = \sum_{j=1}^{k-1} [\frac{gcd(k,i)}{kj}(n-k)]$

$\sum_{j=1}^{k-1} X_k(i,k)[1 + gcd(k,j)\frac{n-i-k-jW_k(i,j)}{kj}$

Onde, $o \leq \leq \frac{dk-1}{gcd(k,j)} - 1$ e
$I_{k,j}(d) = i = ([\frac{dk-1}{j}] + a)j - dk/1 \leq a \leq [\frac{[(d+1)k-1}{j}] - [\frac{dk-1}{j}]$

n/k	2	3	4	5	6	7	8	9	10	11
3	1									
4	1	1								
5	2	2	1							
6	2	1	2	1						
7	3	3	2	2	1					

Aplicação

Temos $n \geq 9$

$\blacklozenge(n) = \frac{n-5}{4} + [\frac{n-5}{12}]$, se $n \equiv 1 \ (mod \ 4)$
$\frac{n-7}{4} + [\frac{n+1}{12}]$, se $n \equiv 3 \ (mod \ 4)$

$\blacklozenge(n) = t(n{-}4, 4, 2),$

Programação

Faça as Importações

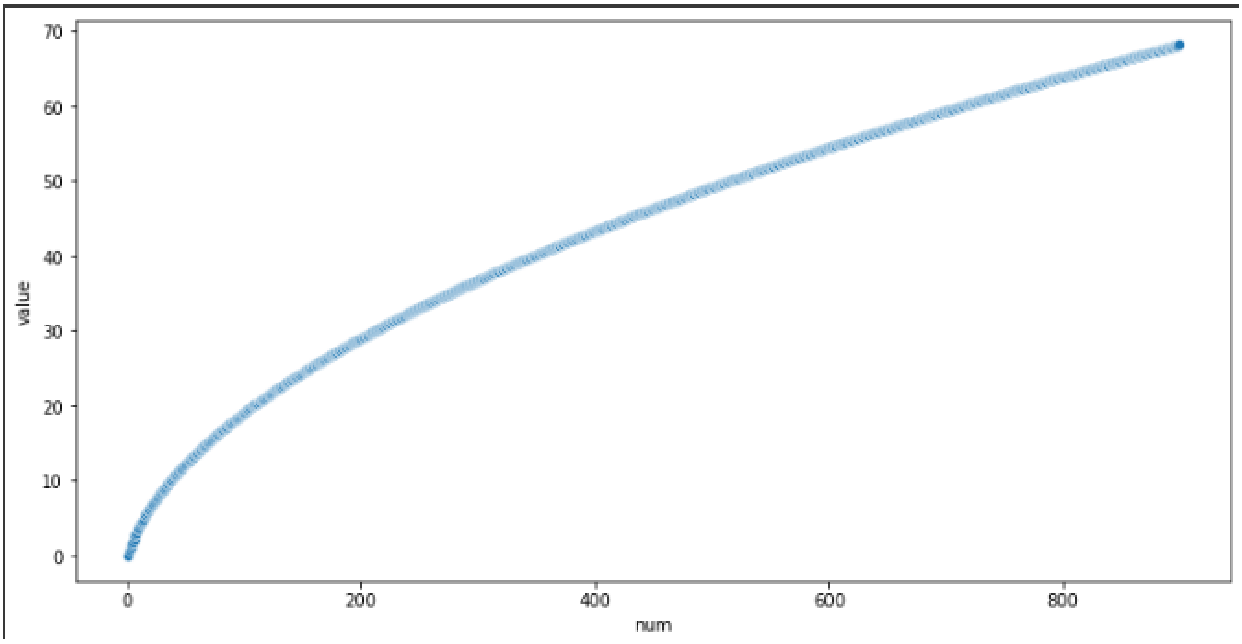
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Leitura do DataSet

```
df = pd.read_csv("sample_data/num2.csv")
```

Plot de Gráfico, do Número em valor do N° de Partições

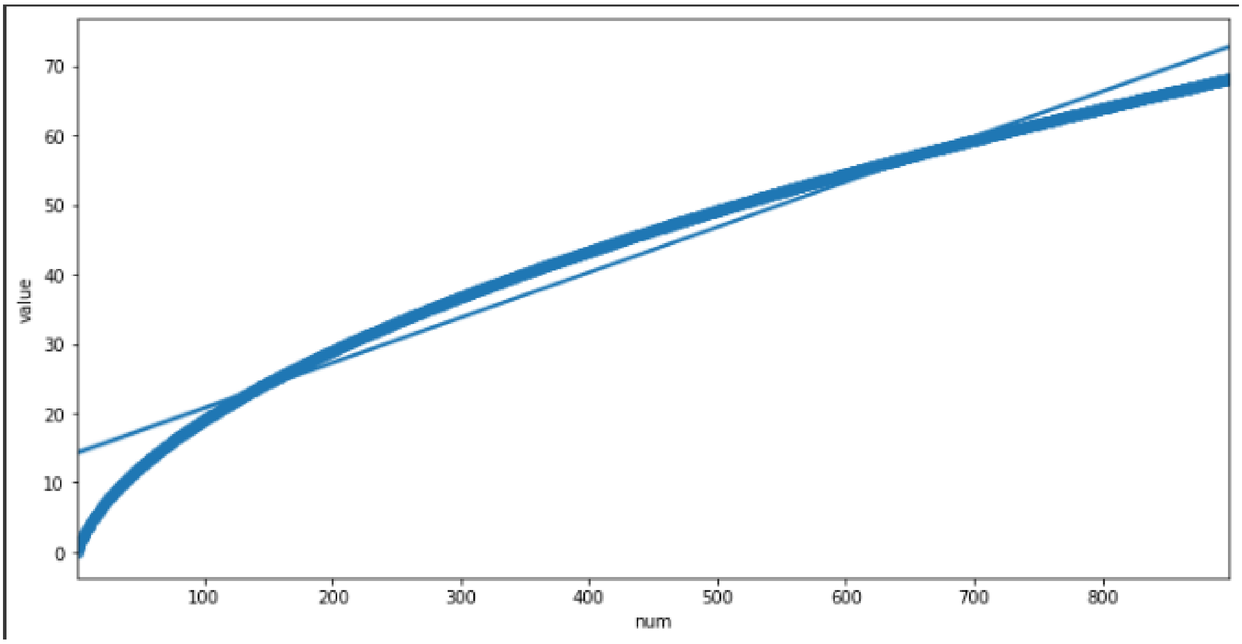
```
plt.figure(figsize=(12,  6))
sns.scatterplot(data=df, x="num", y="value")
plt.show();
```



Este dataset eu apliquei nos valores um log, pois os números neste valor estavam muito grande.

```
plt.figure(figsize=(12,  6))
sns.regplot(data=df, x="num", y="value");
```

Gŕafico com regressão linear



Machine Learning

Importações

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smfmean_absolute_error, mean_squared_erro
```

Divisão entre os valores de x e y

```
x = df.iloc[:,  0].values
y = df.iloc[:,  1].values
x = np.array(x).reshape(-1,  1)
```

Divisão entre treino e teste

```
x_train, x_test , y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Modelo Primário e Treino

```
model = smf.ols('value ~ num', data=df)
model = model.fit()
```

Summary

```
model.summary()
model.params
```

Plot de Regressão Linear

```
sales_pred = model.predict()
plt.figure(figsize=(12, 6))
plt.plot(df['num'], df['value'], 'o')data
plt.plot(df['num'], sales_pred, 'r', linewidth=2)
plt.xlabel('Num')
plt.ylabel('Value')
plt.title('Num vs Value')
plt.show()
```

Teste

```
new_X = 100
model.predict({"num": new_X})
```

Stat Model

```
x = sm.add_constant(df.num)
y = df.value
mod = sm.OLS(y,x)
print(res.summary())
```

Predição

```
pred = res.predict(sm.add_constant(x_test))
print("{:.2f}%" .format(r2_score(pred, y_test)*100))
```

95.96%

$$R^2 = 95.96\%$$

Deep Learning

Importações

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.models import Sequential
```

Modelo da Rede Neural Densa

```
model = Sequential()
model.add(Dense(3, activation='relu', input_dim=1))
model.add(Dense(3, activation="relu"))
model.add(Dense(3, activation="relu"))
model.add(Dense(1, activation="linear"))


model.summary()
```

Compile do Model

```
optimizer = tf.keras.optimizers.RMSprop(0.001)
model.compile(loss = 'mean_absolute_error', optimizer = 'adam',
metrics = ['mean_absolute_error'])
```

Treino com os valores de validação

```
model.fit(x_train, y_train, batch_size=10, epochs=1000, validation_data=(x_
```

Conclusão

Existe uma forma na matemática e na programação para definir a partição de um número inteiro, no entanto eu quis testar se uma rede neural conseguiria encontrar um padrão para o dataset.
Aplicando $log(num)$ os valores ficam bem menores e mais fáceis da rede neural prever.

PS. Este dataset foi desenvolvido por mim